知識情報処理に関する一考察(情報の構造化と意味に関する研究)

AUTHOR(S):

大須賀, 節雄

CITATION:

大須賀, 節雄. 知識情報処理に関する一考察(情報の構造化と意味に関する研究). 数理解析研究所講究録 1984, 525: 170-186

ISSUE DATE:

1984-06

URL:

http://hdl.handle.net/2433/98513

数理解析研究所講究録
525巻 1984年 170-186

知識情報処理に関する一考察

大須賀節雄
（東京大学工学部境界領域研究施設）

1 序

知識情報処理を新しい情報処理体系としてその実像を確定し、また現行の情報処理体系との関係を正しく認識することは今後の情報処理技術を発展させる為に極めて重要である。 知識情報処理は基本概念であるが故に多くの面をもつが、本稿ではこれらの諸面の考察を通して、知識情報処理とは何かを探る。 本稿では知識情報処理を汎用情報処理技術として考察する。

今日、知識情報処理の一応用システムとしてエキスパート・システムが医療をはじめとして多くの分野で開発が進められている。 これは知識情報処理の一形態ではあるが知識情報処理システムそのものではない。 エキスパート・システムは現状では未だ必ずしも広い範囲で実用化されていないし、これを実用問題に適用する際には適用分野を注意深く選定せねばならないのが実情である。 適用する分野の性格に合わせてシステムが定められるという意味で、エキスパート・システムは汎用性を欠いている。 Buchananはこの理由として表1のようないくつかの項目をあげている。 これらが改善されるならエキスパート・システムの汎用性は確実に高まることが期待されるが、問題なのは現在のアークテクチュアのもとでこれらを改善し得るか否かである。 たとえば stylized I/O の問題は知識表現の記述力の問題に関わってくる。 したがってこれを改善するにはシステムの最も基本の部分から再検討を要することになる。

エキスパート・システムにおいて上述のような分野依存の性質が生ずるのは何故なのだろうか。 また、それが避け難いとするなら、少なくともシステムに適合する基準を設定することが望ましいが、この基準をどのように定めるべきだろうか。 これらを知るには知識情報処理という領域と、その中でのエキスパート・システムの位置づけを知り、その上で応用を選ぶということが何を意味するかを明らかにせねばならない。

2 知識情報処理の諸アスペクト

知識情報処理は環境との関係や視点に応じて様々な面を見せる。 これらを注意深く観察し、その結果から知識情報処理のモデルを探ることにしよう。 このような面として以下のものを考える。

（a）知識情報処理の能力と限界

知識情報処理は人工知能のこれまでの成果を実現したシステムである。 しかし、人工知能の成果は知的機能の一部を実現したに過ぎず、このために問題解決

という目的に対しても知識情報処理には限界が生ずる。　知識情報処理がこの面で発揮する能力とその限界を知ることは知識情報処理システムの開発に際しても非常に重要である。

（b）現行の情報処理方式と知識情報処理方式の類似点と相違点

知識情報処理システムは non von-Neumann計算機であるという考え方がある。もしこれが正しいとするなら、実験的にしろ現在の計算機の上で知識情報処理が実施されていることをどのように説明したら良いのだろうか。　また知識情報処理システムが von-Neumann計算機の一部であるとするなら処理方式における相違はどのような意味をもつのであろうか。　これらを明らかにするために現行計算機と知識情報処理の相違点、類似点を明らかにしておくことが必要である。

（c）知識情報処理システムの評価項目

知識情報処理システムの設計に際してどのような点に注意すべきかを考える。

（d）情報の表現レベルと変換機能

情報の変換・処理・蓄積等を広義の情報処理と呼んでおくが、このためには情報表現の形式化が必要である。　この形式によって処理の方式が定まる。　逆に処理方式の相違は情報表現の形式の相違でもある。　この観点から現行の計算機と知識情報処理を比較することは有用でもあり、興味深い。

（e）知識情報処理システムの応用分野

以上の議論を通して、知識情報処理システムの最も的した応用分野を考えることができる。

以下、これら各々についてもう少し立ち入って考察する。

3　知識情報処理の能力と限界－問題解決の立場から

知識情報処理の機械化の大きな目的は知的作業の支援システムを実現することに他ならない。　人の知的作業を広義の問題解決としてとらえ、知識情報処理がどの程度それを支援し得るかを考えてみよう。　それには問題解決というプロセスをモデル化してみるのが効果的である。　問題解決の単純な例として図１を考えてみる。これは自動車のような構造体の振動特性を所与の外力に対する応答の形で求める問題であるが、この問題を解く手順は図のように　（問題）－＞[モデル化]－＞（モデル表現）－＞[求解]－＞（解）－＞[プログラミング]－＞（プログラム）－＞[計算]－＞（結果）

（入力）

となる。　この図で（　　）は情報の表現を、□は操作を示す。　操作は一般に異なった情報表現間の変換として定義される。　この変換には変換前後の情報表現形式に応じて多種類のものがある。　特に知的な作業には知的な機能を含む変換が必要である。　このような知的機能には表２のようなものが含まれる。　たとえば図で[モデル化]という操作には；どのような形式でモデル化するかという発想機能、どのような情報を含めたらよいかといった直勘的判断、かつての経験に基づいて概略を定める類似による連想、具体例からモデル構築を推論する機能、仮想モデルが正しく問題を表現するかどうかについて行なってみる計算などの機能が含まれている。　これら

知的な機能は演算など一部を除けば、それらの機能がどのように働くかというアルゴリズムすらが不明である。

これらの機能のうちで、従来の計算機の上で実現されてきたものは演算を中心とするいくつかの機能までである。　このことから問題解決の全過程において演算等アルゴリズム化され機械で実現されているレベルの作業より上流の部分は人によりなされ、人と計算機との接点は情報表現としてはプログラムの段階であった。　しかし図でここより上流の作業は縦の流れ、これより下流は横の流れで表わしてあるように、この上流と下流では作業の内容が質的に変化している。　いわば上流は構造の合成、下流はその構造のインスタンスの解析とも言うことができる。　この意味で従来の計算機では人と計算機の仕事の分担の境界ははっきりしていた。

一方、知識情報処理はさらに高度の知的機能の実現をめざす。　このことは図１のような単純な例について言えばモデルの段階で人が問題を計算機に渡し、以後は知識情報処理システム側で処理するという分担が成り立つように見える。　演えき的機能を与えることにより、このような新しい接触点を一般論として設立できるだろうか。

これに答えるにはもっと一般的な問題のクラスを考察せねばならない。　問題にはいくつかの異なった型がある。　これを２つのクラスに分ける。　第１のクラスはモデルを固定することができ、そのモデルに基づいて問題の解を求めることができるもので図１の例はこの代表例である。　しかしこのクラスも解という情報表現が存在しているもの、すなわち問題のモデルが数式のような整った言語で表現され、解もその数式で表現できるものと、事務処理のようにそのような解の表現が存在せず、モデルから直接プログラム変換されるものがあり、さらに前者も、モデルにたいする一般解が求まっている場合と、一般解が求まっておらず、モデルに対してその解を求めることが必要なものがある。

第２のクラスは条件（モデルの満たすべき）が与えられて、その条件を満たすモデル構造を求める型のもので、設計、研究・開発のような高度の仕事がこのクラスに入る。　第１のクラスを解析型問題、第２のクラスを合成型問題と呼んでおこう。　合成型問題の解の手順は　（１）仮のモデルを作る。　（２）これに基づいて解析型問題として条件に相当するモデルの性質をとり出す。　（３）これを与えられた条件と比較し、もしこれが条件を満たせば終了し、この時のモデル構造を結果とする。　もし条件を満たさなければモデルを修正し（１）に戻るというもので、中に解析型問題を含んでいる。　当然、合成型の方が問題としては複雑である。　以上をまとめると以下のようになる。

問題の型：

クラス１　解析型　（ａ）解の表現可能かつ解が得られている
　　（ｂ）解の表現可能かつ解が得られていない
　　（ｃ）解の表現なし

クラス２　合成型

図２はこれらの型についてそれぞれが前記の知的機能のうちどれをどの段階で必要とするかを配置してみたものである。

これから判るように、１－（ａ）の型の問題以外はモデル表現以後でも演えき機能以外の高機能を要することは確実と見られ、人と機械との接触点となる。　適切な表

現形式は一般論としてはできる限り上流で問題を機械に渡すことが望ましいので、演えき機能があるにもかかわらず、適切な接触点がないという理由で常に現在の計算機でのプログラム表現レベルに接触点を固定してしまうのは正しくない。　このために実現可能な機能を活用して新しい情報処理方式を確立することが必要となる。　この方式については後に述べることとして以下、2、3の点について注意しておこう。

(1)　上述はマクロな機能に基づいて議論を行なった。　von-Neumann 型計算機は演算機能がさらに単純な機能群に展開され、さらにこれらは最も単純な少数の関数にまで分解されてしまう。　実は演えき機能もパタン・マッチ、バック・トラックなど同じクラスの単純関数で実現される機能で表わすことにより、von-Neumann 計算機の上で実現される。　これが現在の知識情報処理の姿である。

(2)　演えきという機能を von-Neumann型のものと異なる何かの方法で直接（ハードウェアにより）実現することができる。　この時、もし図1で演えきより下流に位置づけた演算等の機能を演えきの機能で表現することができれば、演えきを最低レベルの機能とする計算機の実現が可能になる。　この場合、図1の程度のマクロ機能で演算等の機能が演えきの上位機能に位置づけられることになる。

4　知識情報処理と現行計算機方式の相違点、類似点

現実の知識情報処理システムが演えき機能のみを基本の計算メカニズムとし、他のすべてをこれによって表現する形式をとるかどうかはともかくとして、知識情報処理が演えき機能を表面に出した処理方式であることは事実である。　そこで情報処理の仕組について、知識情報処理と現行計算機方式を比較してみる。

一般に実用問題は複雑であり、複雑な系は例外なく有限個の要素から成る構造として表わされる。　情報処理の場合でも同様に問題の表現の一般形を入力と出力間の関係あるいは関数として表わすと、この関数はもっと簡単な関数の組に展開されてゆき、最終的には極めて単純な関数に達する。　これを基礎関数（もしくは関係表現）と呼んでおく。　基礎関数の組と、構造化規則を与え、これによって問題を表現するのが従来の方式であり、この構造化の逆である分解機能と基礎関数の組の実行機能を備えた機械を実現すればこの問題の評価が可能になる。　知識情報処理についても問題の表現と処理をこれと類似の枠組に合わせて説明することができる。　この両者の相違点と類似点を明らかにすることが目的なので、この同じ枠組で両者を対比させてみよう。　そのため、この枠組を表わす記号を以下のように定義しておこう。

Σ　：基礎関数（または基礎関係の表現）の組

R　：　に適用される合成規則

$F(\Sigma)$　：合成規則を繰り返し適用することにより得られる（合成可能な）あらゆる複合関数（または関係表現）のクラス

σ （$\in F(\Sigma)$）：$F(\Sigma)$ の中の特定の複合関数（関係表現）

$E(\sigma)$　：　の評価機構

＝　$E_1(\sigma)$：$\sigma \in \Sigma$　：基礎関数（関係表現）の評価機構

　　$E_2(\sigma)$：$\sigma \notin \Sigma$　：複合関数（関係表現）の分解機構

4．1　現行の計算機方式の基礎理論

現在の計算機方式の背景となっているのは、チューリング機械という一つの抽象機械に基づいて定義された計算可能の概念とそれを用いて導かれた計算可能な関数のクラスの概念に関する理論である。　この理論のもとで少数の、厳選された基礎関数と、合成規則が定義される。　この表現は必ずしも一様ではないが一例を次のように挙げる。　以下、現行計算機の場合添字$_C$ を付す。

Σ_C：(a) $C_A(x)$：A $\ni x$ なら1，　さもなければ 0

(b) S(x)　：x の次の数値

(c) N(x)＝0 ：常に 0

(d) $U_i^n(x_1, x_2, .., x_n) := x_i$ ，　$(1 \leq i < n)$，i 番目の数を取り出す。

R　：f，g，... h 等が計算可能な時、次の２つの規則で表わされる合成関数 k(x) が計算可能である。　ただし $x=(x_1, \ldots, x_n)$

(a) $k(x)=f(g(x), \ldots, h(x))$，

(b) $k(x)=\min_y[f(x\ y)=0]$　(k(x)はf(x y)＝0 を満たす最小の yを値とする)

この (Σ_C, R) のもとで再帰関数を含む計算可能な関数のクラス $F_C(\Sigma_C)$ が定義される。　ある問題 pが与えられた時、$p \ominus \sigma_C$ により "p とσ_Cは意味的に等価である" を表わすことにする。　σ_C は pのプログラムである。　p が与えられた時、$p \ominus \sigma_C$ なる構造　$\sigma_C \in F_C(\Sigma_C)$を探す（構成する）操作を

$g_C(p : \sigma_C) : \Sigma_C,\ R_C \rightarrow \sigma_C;\ \sigma_C \ominus p$ ，

と表わすと、$g_C(p : \sigma_C)$はプログラミングを表わす。　現行の計算機では g (p: σ_C) は完全に人に任せている。　現実の計算機では理論的な Σ_C に対応して命令語の組が与えられている。

一方、σ_Cが与えられた時、これを基礎関数にまで展開する機能 $E_{1C}(\sigma_C)$と、基礎関数を評価する機能 $E_{2C}(\sigma_C)$は計算機の制御機能、論理演算機能に対応する。現行の計算機における高級言語は Σ_Cから作られる σ_1 ，σ_2，...，σ_m なる有限個の複合関数を新しい基本関数に選び、また R_C の基本的な複合化機能に展開できる複合的な複合化機能を定義して用いる。　これを使う際には、これらの表現系を元の基本系に戻す変換機構すなわちコンパイラが必要である。

4．2　知識情報処理の処理方式

知識情報処理においては知識表現の決定はシステム全体の特性に影響を与える。現実のシステムにおいてはこれにプロダクション・ルール、セマンティック・ネットワーク、述語論理、フレームなど多様のものが用いられているが、述語論理は他の多くのものの解析手段としての機能も果し得るので以下では述語論理で代表して議論する。

論理体系の基本は、一定の形式（述語論理形式）によって表現された論理式（以下、単に式）の集まり　（論理の公理を含む）と、ある表現の組からこれと論理的に等価な別の表現を導く推論規則である。　最も単純な推論規則（モーダス・ポネンス）は、式として A("Aである")と、 A=>B("もし Aなら Bである")が与えられた時、結論としてB("Bである")を導くものである。　これを形式的に A，A=>B/Bと書

く。　この推論規則が前記合成規則に相当する。　これを R_K と表わす。

論理的処理は、ある与えられた記述が、前もって定められた解釈のもとで真か偽かを判定することである。　式集合 Σ_K から出発して推論規則と、Σ_K または Σ_K から導かれた途中結果のみを用いて、与えられた表現 pに到達したら、 pは証明されたといい、この際の表現の列を演えきと呼ぶ。

式集合 Σ_K は、もし、上記の機械的手順により、ある表現とその否定とを同時に生ずる時、矛盾するという。　たとえば Σ_K が、A, A=>B, A=>～B(～B は"Bでない"を表わす）を含むなら、これは矛盾である。　無矛盾の式集合から、推論規則を繰り返し適用して得られるすべての可能な表現のクラスを作り出すことは理論上可能である。

前述の記述に対応して、Σ_K から推論規則を繰り返し用いて導かれるあらゆる表現のクラスを $F_K(\Sigma_K)$ と表わす。　また、推論規則を実現した物理的な推論機構 Dが作られた時、上記 $F_K(\Sigma_K)$ 内の任意の表現 σ_K にたいしてこの物理的機構が実際に演えきを生成することができるなら、この推論機構は機構的に完全であるといおう。

述語論理においては現行の計算機と異なり、（その理論系のもとで許された言語による）問題の表現 pと、p に対応する論理表現 σ_K（すなわち $p \Leftrightarrow \sigma_K$）の間の変換は形式的であり、しかも直接的である。　この間に人による意味の定義が入らないので $p \Leftrightarrow \sigma_K$ と書くかわりに $p = \sigma_K$ とする。　すると機構的に完全な推論機構 Dのもとで

$g_K(p : \sigma_K) : \Sigma_K, R_K \Rightarrow \sigma_K, \quad \sigma_K = p$

すなわち Σ_K と R_K （推論規則）とから、問題への道筋を構成する操作が自動的に行われる。　=>はこの構成操作が自動的であることを示す。

述語論理においては Σ_K が無矛盾であるならそれに応じた $F_K(\Sigma_K)$ が存在する。

したがって、ある問題領域に関する基本情報を Σ_K に加えておけば、$F_K(\Sigma_K)$はその問題領域に関し、前もって準備された情報から証明され得るあらゆる問題のクラスを表わしている。　知識ベースはこれを利用するもので Σ_K が知識ベースを表わすことは明らかであろう。

4．3　問題解決における現行計算機方式と知識情報処理方式の比較

問題を解く基本的な筋道は、このように、プリミティブな表現のクラス Σ と、それから新しい表現を生成する規則が存在して、任意の問題をこの規則のもとで Σ に帰着させることであるという点では現行方式も知識ベース方式も同様であるが、その実現方法に重要な相違点がある。それを列挙してみよう。

(1)　表現とその評価；

C ；基本表現は関数であり、評価はその関数値である。

K ；基本表現は命題であり、評価は論理値である。

(2)　プリミティブの集合；Σ

C ；Σ_C は有限かつ少数で、しかも計算機技術の専門家により厳選されている。したがって Σ_C の無矛盾性が保証されているから、無矛盾性チェックのための特別な考慮は不要である。　他方、実際には Σ_C なる情報蓄積機能があるわけでなく、ユーザは Σ_C を操作できない（ファームウェアとして一部のみ許され

ている）。

K ; Σ_Kは情報蓄積機能としてユーザに開放され、知識の数も一般に多い。 したがってΣ_Kが矛盾する可能性があり、この対策が不可欠である。

(3) 評価機構

C ; すべての $F_C(\Sigma_C)$ にたいし、$E_C[F_C(\Sigma_C)]$ の実行機構が基本要素として（ハードウェアにより）準備されている。 特にΣ_Cのすべてにたいし、E_{IC}の実行機構がある。 関数値評価のための基本は"変換"であり、評価機構が固定的に定義されているため、絶対基準のもとでの評価となる。

K ; 評価は命題の論理値であり、公理（知識）もしくはすでに証明済みの表現との一致によりなされる相対評価であり、その基本操作はマッチングである。Σ_Kがユーザに開放されているから、知識が増せば $F_K(\Sigma_K)$が加速的に増大し、知識をかえれば、評価基準が変わり得るという柔軟性があるので、動的な環境に対応できる。 反面、 前述のように無矛盾性を含め、Σ_Kの管理が大きな問題となる。

(4) 問題表現

C ; 計算機に与えられるのは問題pではなく、それに意味的に等価な（$p \backsim \sigma_C$）評価手順 σ_C である。 意味の等価関係は人が定義する（この定義のもとで p等価な σ_C を人が作る）。

K ; 問題 pとこれに等価な論理表現 σ_Kとは形式の相違があるのみである。 この意味で問題 pが直接、計算機に与えられ、その間に人による意味解釈は必要ない。

(5) 問題解決手順

C ; 与えられた問題 pにたいして解の評価手順 σ_C を生成する機能 $g_C(p : \sigma_C)$は計算機としてもたず、すべて人に任される。

K ; 探索機能を含む推論機構は一般に pにたいして σ_K への演えきを生成する機能 $g_K(p : \sigma_K)$を有する。

5 システム評価項目

知識情報処理の実際の形態を定める上で少なくとも次の4点の検討は必要である。

(1) 応用性 ：適用分野の広さ
(2) 使いやすさ：問題記述のしやすさ
(3) 処理効率 ：処理効率、並列処理の可能性
(4) 既存のシステムとの両立性

(1) の応用性は知識情報処理の存在理由を左右する重要なポイントの1つである。知識情報処理には従来の計算機方式では困難であったり、著しく不便である応用分野に役立つことが期待されている。 このような分野として、設計、研究・開発、診断、意志決定、監視、制御などが挙げられている。 したがって知識情報処理はこれら期待される応用に適するように設計されねばならない。 このた

め、これら応用分野の作業を分析して、そこで必要とする機能を洗い挙げる必要がある。

(2) の使いやすさも重要な要素である。　現在の計算機システムは必ずしも使いやすいものではないし、プログラムの開発が必要であるという欠点を持っている。情報の表現レベルが使いやすさに関わることは明らかで、これが高い程ユーザにとっては望ましい。　ここで次の２点を考慮する必要がある。

(i)　情報表現の最も適切なレベルは個人および応用に依存する。　特定の個人が使い易いと感じるレベルは一様でなく、汎用システムとして不特定のユーザおよび応用を想定する際には、これらの多様性に適合し得ることが必要である。すなわちシステムとして使い易さを保証することは巾広い表現レベルを許すことであり、そのために最上位レベルを出来るだけ高くする努力が必要である。

(ii)　３で示したように、現在の技術では情報表現レベルを高める努力にも限界があり、このレベルまで実現すれば大半のユーザが満足できるものではない。したがってこの範囲内で使い易さを追求する必要がある。　知識情報処理システム開発の主たる努力が情報表現レベルを高めることに向けられているが、上述のようにこれに限界があるという前提のもとで使い易さを追求するのは全く別種の努力になる。　後者のような努力は現実の場で知識情報処理システムがいかに使われるかを応用のパターンを考慮した上でなされる必要がある。

(3) の処理効率の重要性は改めて言うまでもないが、知識情報処理の場合、効率の追求の仕方が従来の計算機と異なってくる。

(i)　並列処理

一般論として情報の表現レベルが高くなるほど、それを処理するまでに必要な変換の過程も増し、また探索型の処理が入ってくるので効率は低下する。　情報の構造化、推論の効率化などソフトウェア面での努力は当然であるが、最後はハードウェア化による他なく、特に並列処理化が重要である。

並列処理が効果的に行なわれるか否かは、処理対象となる情報に内在する逐次性にある。　この逐次性は (a)問題の意味に関わる部分（時間的順序、概念間の依存関係等）と、(b) 処理方式上生ずるものがある。　並列化を進めるためには後者を減少することが必要であり、これは再び情報表現の問題に還元される。　すなわち、高レベル表現は (a)以外の逐次性を含まないものである。

(ii)　基本表現レベル

知識表現レベルを基本の表現レベルとすると他のすべての関数や関係はこのレベルの表現からの合成として表わされねばならない。　特に演算関係の操作を、知識の表現レベルで表わさねばならない。　これは計算機械上で扱われる数値の集合に関しては不可能ではない。　ユーザにとって直接関心があるのはまず情報表現のレベルであって、これが同一であるなら内部表現はいかなるものでもよく、効率の良いものが選ばれる。　処理効率がこのようにして情報処理システムの情報レベルの決定に関わってくる。

(4) 既存資源との両立性

現行計算機システムは広く普及している。　もし新しく設計されるシステムが従来のものとの並立性がないと、既存のものはすべて入れ替えられねばならない。

問題なのは、これまで蓄積されてきた情報資源との両立性であり、既存情報（プログラム、データ）が有効に利用できない場合、新しいシステムへの移行が困難になる。　これも情報表現の問題として考えられねばならない。

これらを考慮して、知識情報処理の情報表現レベルの設計を行なう必要がある。

6　情報表現のレベルと変換機能

6．1　情報表現のレベル

3節において、人と計算機との接触点における情報表現のレベルの概念を必要とした。　また4節において、問題の表現 pと、計算機内でそれに対応する表現　の間に意味的な変換が含まれる場合と含まない場合があることを述べた。　これらの問題を説明するために、ここでさらに情報表現のレベルに言及する。

一般に情報の表現は人に理解しやすいものほど高いレベルであるといわれる。　逆に機械が理解（処理）しやすい形式は低レベルといわれる。　4節の情報処理の枠組の議論において、問題の記述は基本関数（関係）の複合化、複合表現を処理するにはこれの単純化（基本表現の方向へ変換する）機能が必要であることを示した。　ここで複合表現は問題表現に近く、高レベル表現であり、基本表現は機械と結びついた低レベル表現である。　このことから (1)情報には多くの表現レベルがあること、と (2)高い表現レベルを低い表現レベルに落とすために両レベルで定まる一定の機能があること、を情報の表現と処理の一般的枠組としてもよいだろう。　この観点から現行の計算機と知識情報処理の処理方式を表わしてみよう。

図3は現行計算機内の情報レベルとその間の変換機構を示したもの、図4は知識情報処理とその特徴部分を示したものである。　4節で示したように、現行計算機では問題の表現 pと、その計算機内表現 σ_c の間に意味的変換が必要であり、それを人が行なっていることから、問題の表現自体は現在の人と計算機の接触点であるプログラム・レベルより上方にあり、それを点線で示している。　これに対応して $p \Leftrightarrow \sigma_c$ の関係が示されている。　同図にはさらに参考として、計算機内の低レベル表現に対応する外界の情報表現レベルの例を併記している。

図4は知識情報システムの基本部分のみを示しているが、現実の知識情報システムの形式を定めるにはさらに多方面にわたる検討が必要であり、この図のように単純なものではない。

6．2　知識情報処理方式と現行計算機方式

現行計算機方式（図3）と知識情報処理方式（図4）の関係はこの情報表現レベル構成で考えると比較的明確である。　プログラム・パッケージを内部は見ずに入力と出力の関係としてプログラム・パッケージ名と共に述語のようにみなせばこれは推論の対象として知識表現の一つとみなせる。　逆にプログラムはこの述語の評価機構と考えられ、推論機構がこれを呼び出す構造をとることができる。　この時の関係は図5のようになり、知識表現がプログラムの一つ上のレベルということになる。

ここで知識情報処理システムの構成に際して２つのケースが考えられる。　第１はシステムを実際に図５のように作ることで、従来の計算機の上に推論機構を載せる。

さらに推論機構の実現方法にもソフトウェアで実現する場合と、これをハードウェア化するものがある。　一般にプログラム・レベルの表現を用いてさらに上位の機構（アルゴリズムが見出せれば）を表現できる。　現状の多くのシステムはソフトウェアでこれを実現している。

第２は図４のように知識表現のレベルを最下位とし、現行方式との連続性をもたないものである。　この方式は従来の計算機とは全く異なるシステムを作ることになるが、数値演算が困難（不可能ではないが）なこと、従来の資源の利用は不可能になることなどが問題である。

６．３　上位レベルの表現の可能性

図４でプログラム・レベルの上におかれたレベルは様式化された知識表現レベルであり、問題表現のレベルではない。　図１で示したように原問題の表現は機械処理が困難であり、これを機械的な処理のできるレベルの表現まで下げるには現在われわれの主註にある機能では不足である。　図５ではこの部分を[?]で示してあるが、これは単一の機能ではなく問題レベルと様式化された知識レベルの間には未だ多くのレベルが存在し得る。　たとえば、もし帰納的推論が実現出来たとしたら、データの知識化や知識の整理が可能になり、表現も少なくともデータと知識の混合が許されることになろう。

しかし帰納的推論についても未だ一般的に利用できるようになるまでには時間がかかる。　現在必要なことは、演えき機構までが実現可能という前提のもとで前記システム評価項目を参照しつつシステムの設計を行なうことである。　これを行なう前にもう少し準備をしておく。

６．４　知識レベルとプログラム・レベルの相違

知識レベルをプログラム・レベルの上位に位置づけたが、この両者を区別する特徴は何かを明確にする必要がある。　４で述べたようにプログラムによる表現 σ_c と問題の表現 pとの間の変換には人による意味の等価関係の定義、すなわち”人の与える意味と等価な機械の動作”を定義する必要がある。　このように作られたプログラムは情報の持つ本来の意味と機械に固有の性質、との混合した表現である。　知識表現には機械に固有の性質は入ってこない。　機械の性質の中で最も特徴的なものは処理の逐次性である。　このために、複雑な表現を要素の構造として表わす時、要素間に本来の意味と無関係に順序関係が入る。　これがしばしばプログラムの作成を非効率的なものにし、かつ並列処理をしにくくしている。　したがってプログラム・レベルより一段上位の知識表現がプログラム表現に対して持つ最大の特徴は”処理条件を表現の中に含まない”こと、あるいは”処理システムと全く無関係に本来の意味表現を表わすことができる”ことである。

なお、プログラム・レベルにおいては機械語の他、各種のプログラミング言語が開発されている。　後者は高レベル言語と呼ばれる。　これらすべての高級言語は上述の定義から、全体として知識レベルまでは高級化していないが機械語のレベルに比べ

れば局所的な意味で高レベル化が行なわれている。　この意味でプログラム・レベルと知識レベルの間には多くのサブ・レベルが作られている。　しかしこれらは知識レベルには達していないという点ですべてプログラム・レベルとされる。　したがって知識情報処理システムの設計で第1に心がけるべきことは上述の意味で知識レベルに到達している知識表現言語を設計することである。

7　知識情報処理システム応用の型とシステムの条件

以上の議論を考慮して知識情報処理システム設計の基本として
(1) 完全にプログラム・レベルより1レベル高い知識表現を設計する。
(2) 知識レベルは問題レベルとは異なるので、このレベルで使い易いシステムを設計する。
が導かれる。　使い易いという意味をもう少し break down するにはこのような知識表現をもつシステムの様々な応用を考え、その使い方を考えることが必要である。
知識情報処理システムの典型的な応用の型として、次の2種類を考える。
(A) 質問応用型
(B) 問題解決型

7．1　質問応答型

システムの外部で発生する要求に応じて知識ベースを探索し、そこから要求にたいする答を推論を通して導き出し要求に応答する型の問題で、データベースや情報検索の一つの発展形式である。　情報検索との基本的な相違はキーとの直接マッチングのみでなく推論を通して検索が行なわれることにあるが、情報検索でもシソーラスの利用は限定された推論機能といえる。　むしろ情報検索システムを質問応答型の知識情報処理システムの一形態とみるべきでありデータベースに関しても同様のことが言える。

すると知識情報処理システムは情報検索システムやデータベース・システムに含まれている問題を引き継いでいることになる。　これらは知識情報処理を将来、汎用情報処理技術とするためには解決してゆかねばならないものである。

[A] 情報検索

情報検索システムの評価尺度に適合率もしくは損失率（1－損失率）と雑音率などがある。　これを次のように定義する。　ユーザはキーワードの組σを与えたとする。　システム内に貯えられているすべての情報の中でこのσに丁度適合する情報の組をA とする。　しかし現実のシステムでは Aが求まるとは限らず、実際に得られるのはこれとは少しずれた情報集合 Bである。　集合 X (X :A or B)の濃度（要素数）を |X|とする。　また　l：損失率，　n：雑音率　とすると

$l = |A-B|/|A|$,　$n = |B-A|/|A|$

である。

情報検索システムでは l, n を 0にする保証はないが、その理由は
(1) ユーザとインデクサ間の語の意味や用語法にたいする解釈の相違

(2) 検索機構の不完全性

による。

この問題は知識情報処理でも生じる。 ただし、情報検索システムでは (1)をシソーラスの充実により減らす方向にあり、シソーラスとその処理形は限定的な知識ベースと推論機構と考えられるから、これを体系的に持つ知識情報処理システムでは用語の意味定義を知識として与えることによりこれを減少することができる。 しかしそれには推論機構が多用されるから (2)の機構上の不完全性があると、これの影響が増大する。 したがって知識情報処理システムとしてはこの機構面の完全性を保証することが必要になる。 これは推論の完全性の問題として述語論理で定義されているが一般に知識表現と推論アルゴリズムに依存する。 これには知識の集まり Σ_K と推論規則 R_K のもとでこの体系全体の振舞いを理論的に解析する必要があるが、現在の多くのエキスパート・システムではこのような体系全体としての理論解析はなされておらず、完全性の保証はない。 このような体系を持つのは述語論理のみである。 述語論理では

完全性： $\Sigma_K \models \sigma_K \rightarrow \Sigma_K \vdash \sigma_K$

健全性： $\Sigma_K \vdash \sigma_K \rightarrow \Sigma_K \models \sigma_K$

で定義される完全性と健全性が証明されている。 ここで Σ_K は述語論理のシンタックスのもとで論理式の形で表わされる知識の集まりで、 σ_K は同じく論理式で表わされた質問とする。 $\Sigma_K \models \sigma_K$ は Σ_K と σ_K の同義性を表わしており、 $\Sigma_K \vdash \sigma_K$ は Σ_K から推論の手続きによって σ_K が導かれることを示す。 したがって上記、完全性、健全性は同義性と検索可能性が一致することを示す重要な定理である。

一方、データベースに関してはデータ依存性と整合性の問題が残されている。 これらも個々のデータ単独でなく情報間の関係の問題であり、近年は述語論理で扱われる傾向にある。 依存性も整合性も、その条件を述語論理で表わし、その集まりにたいして、データまたはデータの集合が容認され得るか否かを推論を用いてチェックする。 条件に合うデータベースが存在し得るか否か（アームストロングの問題）は ” が無矛盾であればモデルが存在する”（ゲーデル・ヘンキンの定理）が基本的な保証を与えている。 知識が大量になった時、人が Σ_K の無矛盾性をチェックすることは困難になる。 そのためこれをシステム自身がチェックできる方式とする必要がある。

以上のように知識表現はプログラム・レベルより完全に 1 レベル高いものであると同時に、推論の完全性、健全性を保証しかつ無矛盾性の保証を与え得るものであることが必要である。 ただし、無矛盾性の問題はある一つの知識体系についての問題であり、一つの知識システムが Σ_{Ki}, $i = 1, 2, \ldots, N$, のように N個の知識体系をもつことができる。 $i = j$ のとき $\Sigma_{Ki} \cup \Sigma_{Kj}$ は矛盾を含んでいてもよい。 ただしこの時、ある質問 σ_K にたいし Σ_{Ki} と Σ_{Kj} が同時に（混合的に）用いられることはないように制御する必要がある。

なお、質問応答型のシステムの特徴を次のように表わすことができる。 無矛盾の知識の集まり Σ_K が与えられた時、 $\Sigma_K \vdash \varphi$ なる式 φ を Σ_K に加えるという操作を続けることによって得られた拡大知識系を $F_K(\Sigma_K)$ とする。 この時、知識 Σ_K と推論機構をもつ知識情報処理型質問応答システムは $F_K(\Sigma_K)$ をもつ情報検索システムと

同様であり、解答として得られる情報は必ず前もって準備されている。　すなわち $F_K(\Sigma_K) \ni \sigma_K$ なる情報を与えるのみで、情報を生成する機能をもたない。

[B] 問題解決型

問題解決型の応用は与えられた問題にたいし Σ_K と矛盾しない表現 σ_K を見出すことである。　$F_K(\Sigma_K) \ni \sigma_K$ の時は質問応答型として [A]と同様である（3で示した解析型（クラス1)(a)型）から、問題解決型の特徴的なものは $F_K(\Sigma_K) \not\ni \sigma_K$ の場合である。

このような σ_K を見出す一般的な手順は図1で示した発想、帰納的推論など高度の機能であって、現在ではそのアルゴリズムは未知のものである。　結局人の能力に頼る他ない。　知識情報処理システムとしてはいかにこれを支援し得るかに知識情報処理方式の存在理由があるといっても過言でない。　特に設計、開発のような場合、σ_K は複雑な構造をもったモデルであり、これを見出すことは人でも容易でないし、現実には多数の人が協力してこれを行なっている。　現状ではここに多くの問題（相互のコミュニケーション、整合性等）を含んでおり、知識情報処理が効果を挙げる可能性は大きい。

設計では σ_K は基本的な要素（問題により実体、関数、関係がある）からの構造体として表わされ、これを構築してゆく。　この構築をしやすくするために必要なことは、このモデルの多くの部分が他の部分と独立で、多くの人がそれぞれの部分を受け持ってモデルを作ってゆき、しかも全体としてモデルの整合性が保たれていることである。　これはモデルの表現の問題であり、知識レベルまでレベルがあがっていればこれを用いて上記の性質をもつモデル構築用言語の設計が可能となる。　これが今後の課題であろう。　このためには知識表現の一定の記述性を必要とする。

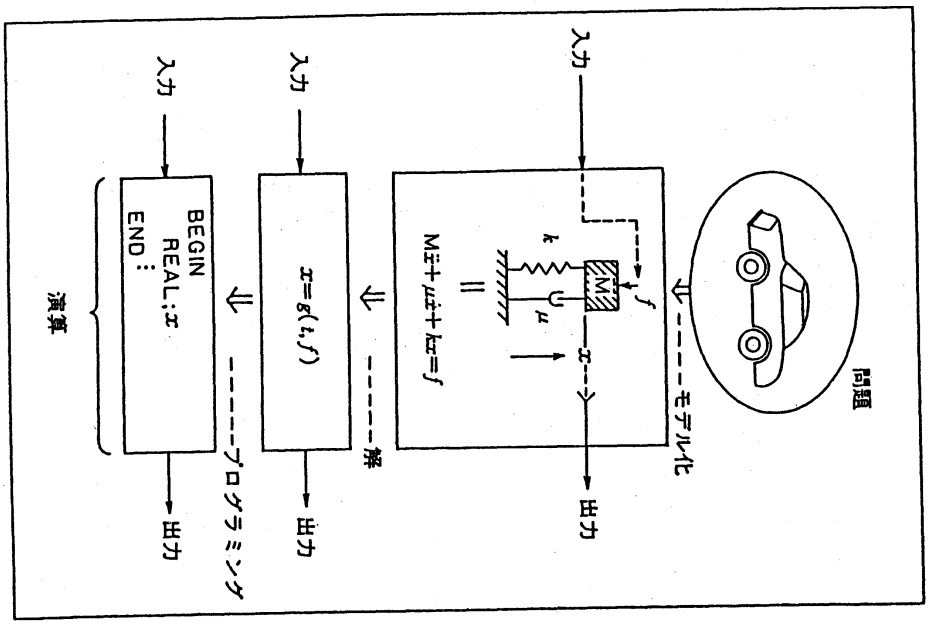
8　むすび

以上、知識情報処理の全体を見通すことによりその実像を明らかにすることを試みた。　以上を通して言えることを要約すると、

1　知識情報処理の目的とは情報表現レベルを現在のレベルより1レベル高めることである。

2　努力目標としては (1)完全にそのレベルに達した知識表現を見出すこと　(2)それでもなお問題表現レベルよりは低いので、その知識表現を適切に用いて問題解決を支援するシステムを開発すること　(3)知識表現としては (1)に加え、推論の完全性、健全性や無矛盾性を保証するものであること、モデル表現が可能なだけの記述力をもつことが必要である。

図1　問題解決のある手順の一例



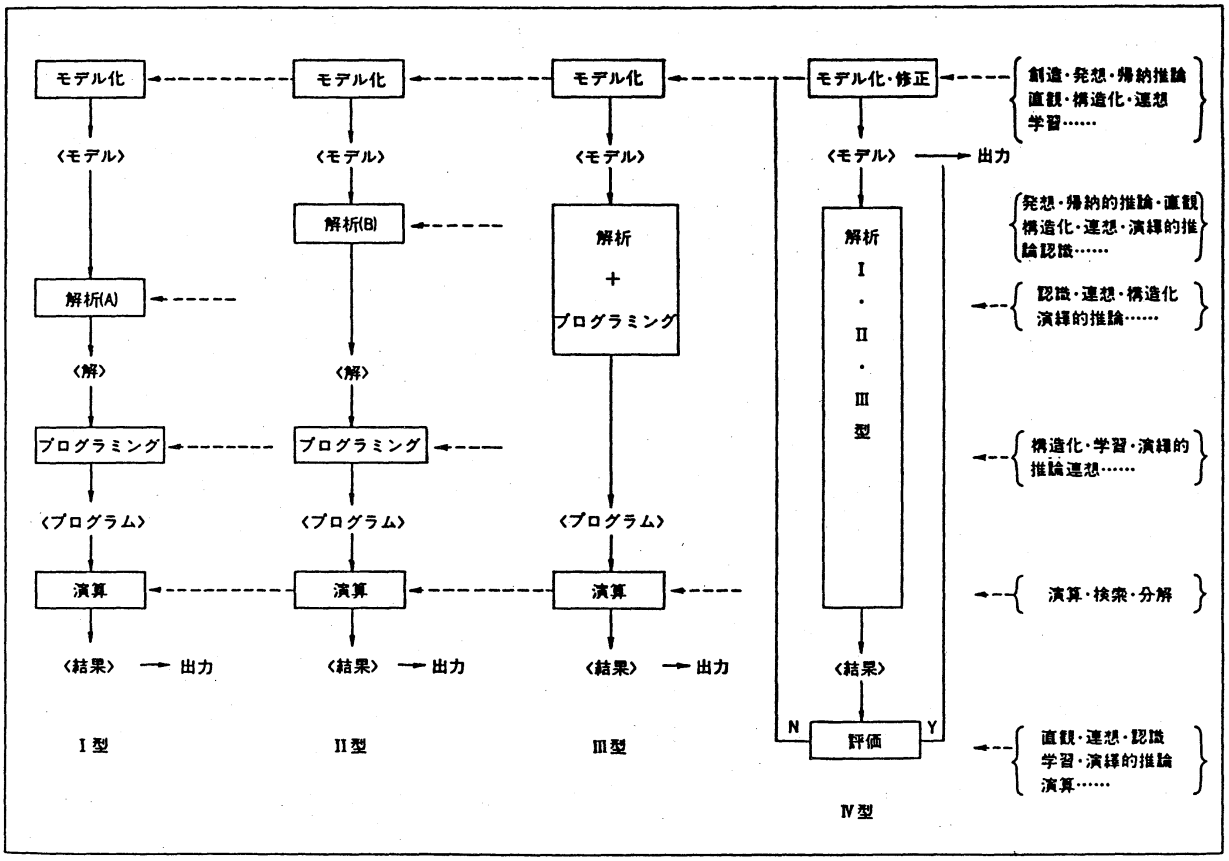


図2　情報処理の各種パターンと必要な機能

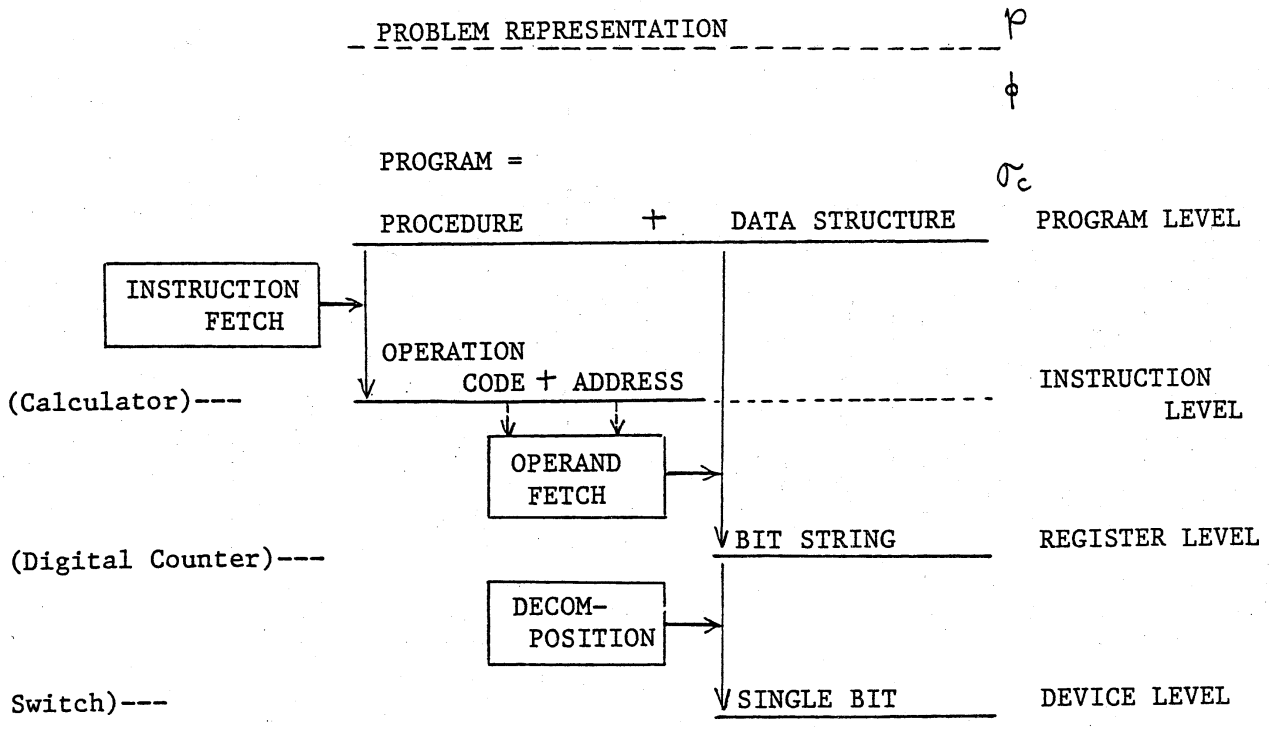


Fig. 3 Infomation levels in current computer systems

図3 現行計算機内情報レベル

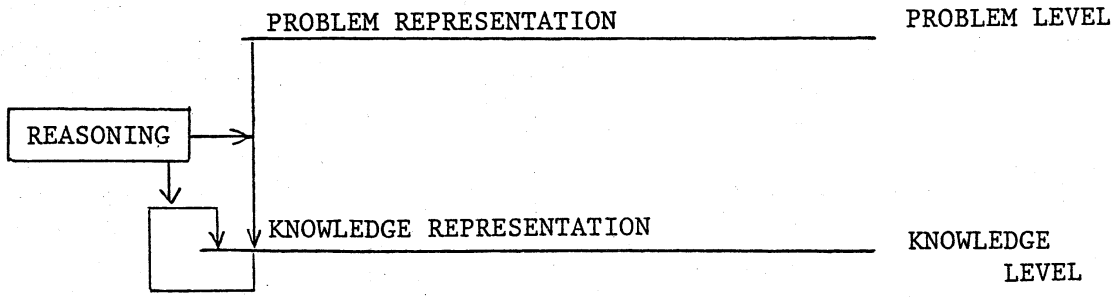


Fig.4 An assumed structure of information levels in knowledge information processing systems

図4. 知識情報システムについて想定された情報レベル

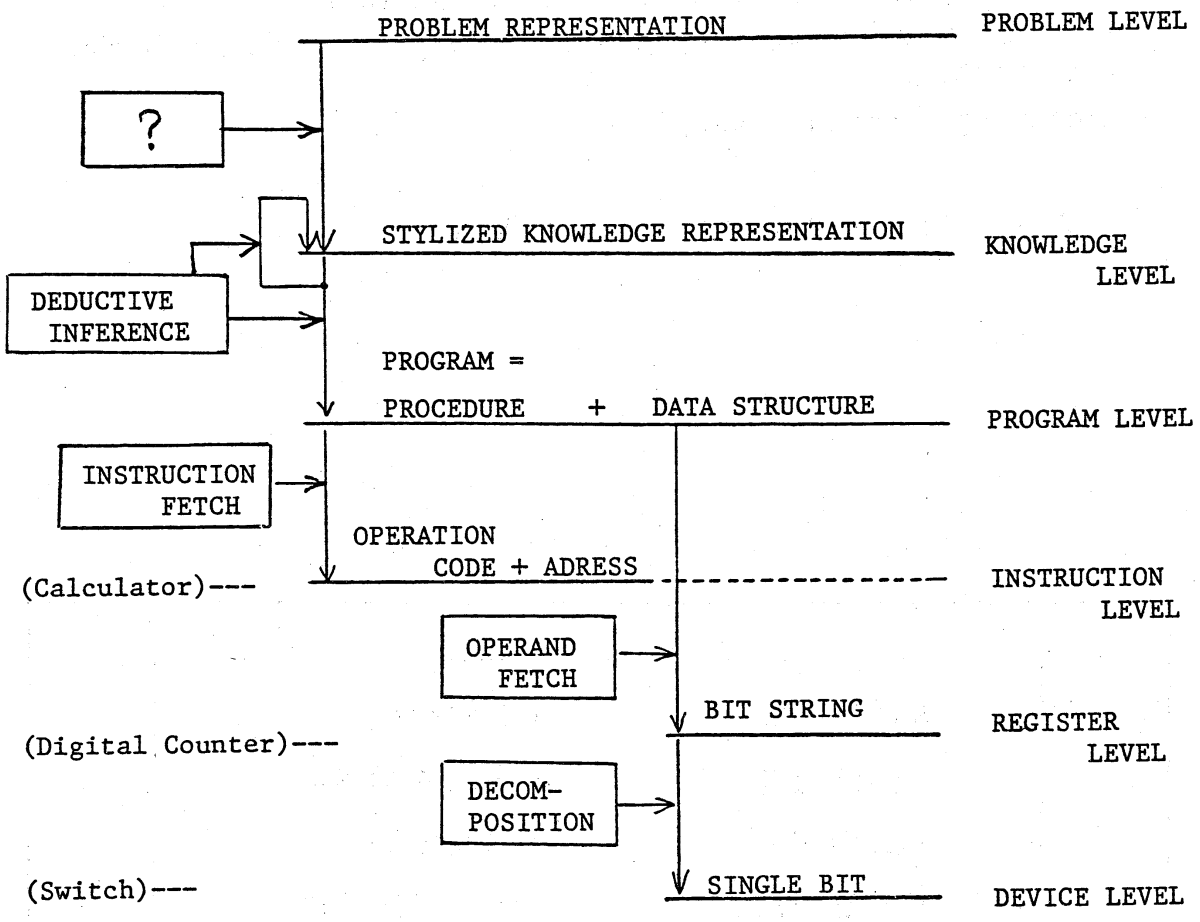


Fig. 5 Infomation levels in real knowledge information processing systems

図5 知識情報処理システム内の情報レベル

- Narrow domain of expertise
- Limited language for expressing facts and relations

- Limited assumptions about problem and solution methods (help required from a knowledge engineer)
- Stylized input/output languages
- Stylized explanations of line of reasoning
- Little knowledge of their own scope and limitations
- Knowledge bases extensible but little help available for initial design decisions
- Single expert as "knowledge czar"

表 1. エキスパート・システムの反省点

- 創造機能
- 発想機能
- 認識機能
- 連想機能
- 構造分解機能
- マッチング機能
- その他
- 学習機能
- 帰納的推論機能
- 構造化機能
- 演繹的推論機能
- 検索機能
- 演算機能

表 2. 各種機能